Section 19
The Alert Server Processor Function

Purpose

The *Alert Server Processor* (*ASP*) provides traffic alert reports generated by the *Traffic Demands Database Processor (TDB)* to the *Traffic Situation Display (TSD)* to be used in Monitor/Alert. The *ASP* also transfers the alert data to field sites via *Network Addressing Communications.* See Figure 19-1.

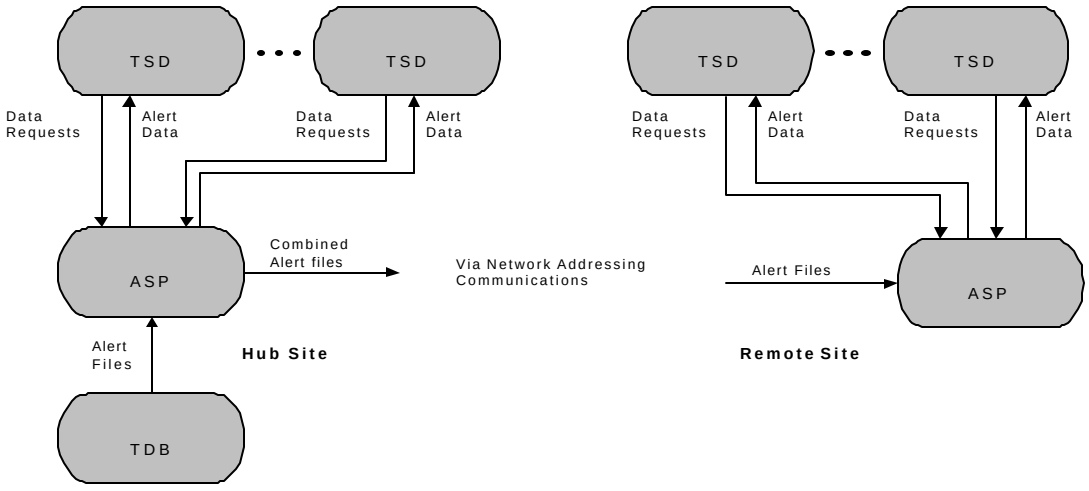

Figure 19-1. Data Flow of the Alert Server Processor

Execution Control

The *Alert Server Processor* is a continuous process that is started by the run-time support process called *nodescan*. If the *ASP* fails, it will be restarted by *nodescan*.

Input

The *Alert Server Processor* receives two arguments at startup. The first argument is the name of the *ASP* configuration file. This file contains various parameters needed by the *ASP*. The configuration file contains the following information:

(1) Local (**L** or **l**) or remote (**R** or **r**) indicator - This parameter specifies whether the *ASP* is running at the hub/local or field/remote site.

(2) *TDB* Provider class name in the format: !class (Optional - uses default of *TDBEV* if not provided.).

(3) Primary site on which to register to *TDB* in the format: $site.

(4) Secondary site on which to register in format: $site. (Optional)

(5) Default string name to send data in format: *string.

(6) All possible strings names in the format: $site //node. Each string name can be followed by a list of all site and corresponding node names to send the data.

NOTE: Items 2-6 are valid only for hub site *ASP* (local).

The second argument is an optional archive flag in the format: **archive** or **ARCHIVE**.

All other input is received through the *Node Switch* as follows:

(1) Alert data files - The *Traffic Demands Database Function* (*TDB*) sends the names of the alert files to *ASP*. There are three element alert files produced for the sector, airport, and fix types.

(2) *TSD* alert data requests - The *TSD* sends messages to the *ASP* containing the element name, element type, and request type.

(3) Messages from the *Node Switch* - These messages consist of statistics requests usually made from *Net.Mail* processes or file transfer messages.

Output

(1) Alert files - The *ASP* processes the element files received from the *TDB* into a file referred to as the **tmd** file. This file is sent to remote *ASP*s via the *Network Addressing File Transfer Process (FTP)*.

(2) Archive file - If the archive flag is provided on startup, the *ASP* will create archive files in the /asp directory. An archive file is created for each airport,

sector, and fix alert report received from the *TDB*. These files consist of the data from the alert files in an ASCII format to be used for data analysis. Refer to section 19.2.2 for file name format.

(3) Responses to *TSD* requests - These messages to the *TSD* contain the requested element information.

(4) Global time line files - The name of the global time line file is sent to the *TSD(s)* whenever an *TSD* connects to the *ASP* or a new file is created, or bar chart request is made.

(5) Messages to the *Node Switch* - The messages to the *Node Switch* mailbox consist of statistics data or the names of the **tmd** files to be transmitted.

(6) Error file - The *ASP* creates an **alert_server_log** file that contains error and message logging information.

Processing Overview

The *Alert Server Processor* consists of three primary functions: *Network Addressing interface for file transfers and communications*, *alert data maintenance and reformatting*, and *alert data service provider for the TSD*. The *Network Addressing interface* handles the receipt of data from the *TDB* and the distribution of the data to remote *ASPs* as well as statistics reporting. The *Alert Data maintenance* is used to process the *TDB* files and maintain the data for distribution. The *Alert Data service provider* is used to provide the alert information to the *TSDs*. See Figure 19-2.

The hub/local *ASP* connects to the Network Addressing node switch process and then requests data from the *TDB Evaluation Server* process (*TDBEV*). To request the data, *ASP* sends a registration message to the *TDBEV* for alert data. After a successful registration is made, the *TDB* sends alert file names to the *ASP* once every five minutes. The alert data consists of three files **tma**, **tms**, and **tmf** for airport, sector, and fix alerted elements. After receiving the alert files, the *ASP* processes the file and stores portions of the alert data in an internal alerted element list. The *ASP* also combines the data from the three files into a single **tmd** file, and stores it in the */asp* directory. The hub site *ASP* will then transfer the **tmd** file to a list of remote sites via Network Addressing. The **tmd** file is stored in the */sio_files* directory on each remote site. Refer to section 19.2.8 for file name format.When the file is received by the remote *ASP*, the data is then processed and stored in the internal alerted element list. Regardless of whether the *ASP* is local or remote; every time an alert file is received, a file containing a subset of the data in the alert files, known as the global time line file, is generated. This file is then sent to a list of registered client processes generally *TSD*s. The *TSD*/client process requests the data in the same method that *ASP* requests *TDB* data. In addition, the clients can request additional data known as bar chart files.

Error Conditions and Handling

The following error conditions are handled by the *Alert Server Processor*:

(1) If an error occurs in obtaining the ETMS default path, the process exits.

(2) If no configuration file is provided, the process exits.

(3) If an error occurs in creating the **tmd** file or bar chart files, the process exits.

(4) If a fatal error occurs, a cleanup handler is called to un-register all connected clients, close the registration to *TDB*, close the node switch mailbox connection, and close all open files.

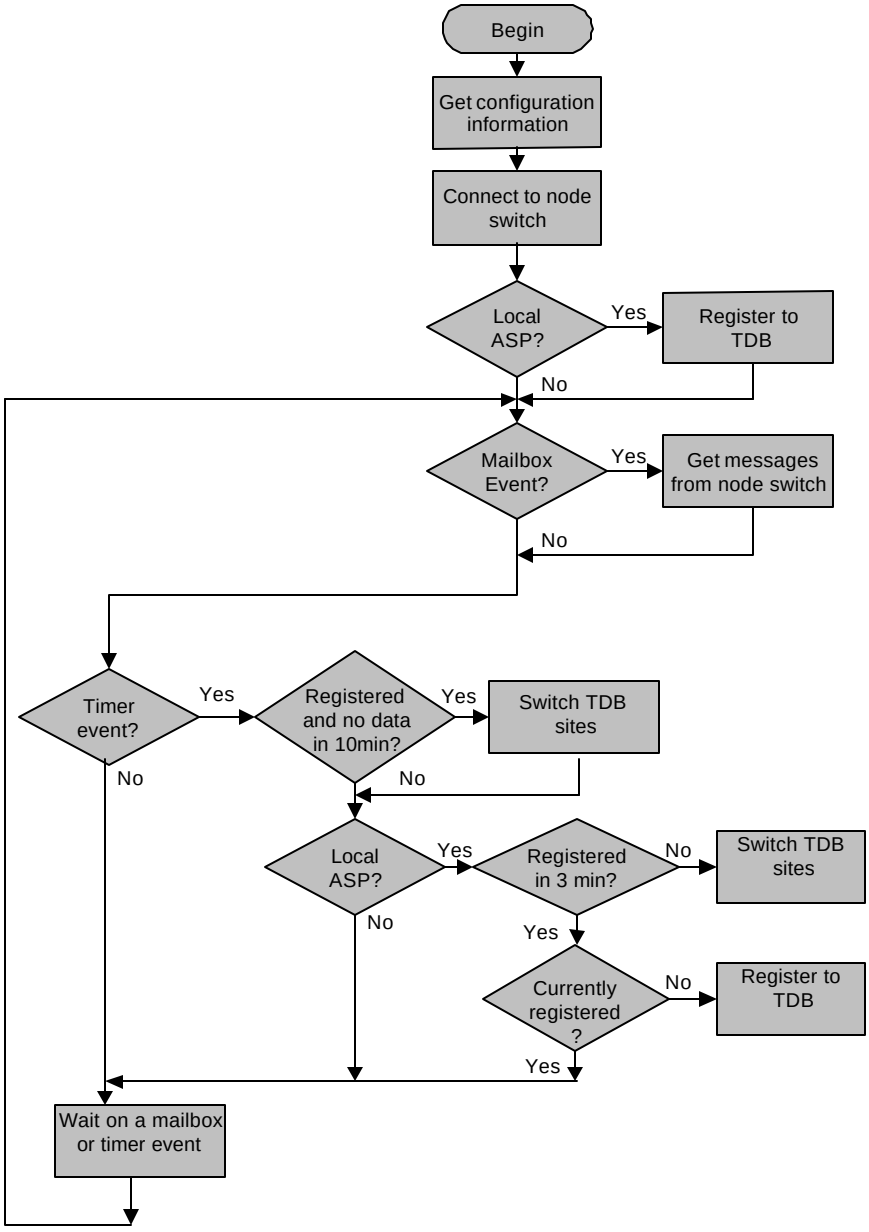

Figure 19-2. Main Logic Flow for the ASP Process

19.1 The Network Addressing Interface Function

The following routines are involved in the Network Addressing interface.

19.1.1 check_node_switch

Purpose

This procedure checks the node switch mailbox for any incoming messages.

Input

ETMS messages.

Output

None.

Processing

The node switch mailbox is polled until there are no more messages. The following messages are received:

- returned messages
- site re-connection - This message indicates that the node switch has reconnected to the site switch. If this message is received, *ASP* will reset its address and register to the site switch as a service provider.
- registration acceptance - This message only applies to the hub site *ASP* process. It is received from the *TDB*, and *ASP* verifies that the message has been received from the *TDB* at the active site. If not, *ASP* will close the registration.
- registration bad - This message only applies to the hub site *ASP* and indicates an unsuccessful registration attempt.
- registration closed - This message only applies to the hub site *ASP* and indicates that the *TDB* process is exiting.
- data message - This message only applies to the hub site *ASP*. It is received from the *TDB* and contains the name of the alert file (**tma**, **tms**, **tmf**).
- file ok - This message is received from the FTP process and indicates a file has been successfully transmitted.

- file bad - This message is received from the FTP process and indicates a file transfer has failed.
- registration request - This is a message from an *TSD* indicating a request for services. If the service count is zero, it indicates the *ASP* process is exiting and should be removed from the list.
- reconfigure - This is a request from a user for the *ASP* to read a new/modified configuration file. It can also contain a new primary/secondary *TDB* site name.
- statistics requests
- *TSD* acknowledgement
- bar chart request

All other messages are returned to the sender as an error.

Error Conditions and Handling

If an error is received in checking the node switch mailbox, the connection is closed and a new connection attempt is made.

19.1.2 close_node_switch

Purpose

This procedure closes the connection to the node switch mailbox.

Input

None.

Output

None.

Processing

This procedure is called when an error is received in sending or receiving a message from the node switch. It sets the connection handle to nil.

Error Conditions and Handling

None.

19.1.3 get_new_ecs

Purpose

This procedure obtains new event counters when a node switch re-open has occurred.

Input

None.

Output

None.

Processing

The node switch initiates a mailbox channel close and re-opens when it has detected 2^{30} bytes of data through the channel.

Error Conditions and Handling

If an error occurs in obtaining the event counters, the node switch connection is closed.

19.1.4 get_site_ids

Purpose

This procedure obtains the site identifiers based on the ASCII site names specified in the configuration file.

Input

None.

Output

None.

Processing

This procedure is called only for the hub site *ASP*. The *ASP* receives a list of remote site names to receive the **tmd** files. The site identifiers for each site are requested via the node switch.

Error Conditions and Handling

If the site identifier is not available or an error occurs in receiving it, the value is left as nil. The *ASP* will re-attempt to obtain it later.

19.1.5 initialize

Purpose

This procedure performs all *ASP* initializations.

Input

None.

Output

None.

Processing

The internal site table and message statistics are initialized. The ETMS default path and program arguments are obtained. The error file and working directory (/asp) are created. The configuration file is read. If the *TDB* class name is not provided, the default of *TDBEV* is used. The node switch mailbox is opened. If the node switch connection is made and the *ASP* is local, the site identifiers are obtained.

Error Conditions and Handling

If the ETMS default path is not found, the *ASP* will abort. If the configuration file name is not provided as an argument, the *ASP* will abort. If an error occurs in creating the error file, the messages are sent to standard output.

19.1.6 open_node_switch

Purpose

This procedure attempts to connect to the node switch mailbox.

Input

None.

Output

None.

Processing

An attempt is made to open the node switch mailbox. If successful, a message is sent to the site switch to register the *ASP* as a service provider for the site. If this is a hub site *ASP*, an attempt is made to register to the *TDB* process.

Error Conditions and Handling

If the connection attempt fails, a timer event is set for one minute to retry the connection.

19.1.7 process_ftp_message

Purpose

This procedure processes a message from FTP indicating a **tmd** file has been received. This applies to remote *ASP*s only.

Input

FTP message, which includes the source filename, destination filename, file size, file type, and sender address.

Output

None.

Processing

The *ASP* first validates that the file received contains /tmd in the destination path. The site name is removed from the destination path and the file is then processed. If the file is processed successfully, the file is renamed, and a global time line file is produced and distributed to all registered processes (*TSD*s). If the archive flag is set, archive files are then created.

Error Conditions and Handling

If the file is not a **tmd** file, it is ignored by *ASP*.

19.1.8 process_reconfigure

Purpose

This procedure processes a reconfigure command to either read a new configuration file or change the primary/secondary site names for the *TDB*. Only the local *ASP* will accept a reconfigure command.

Input

The address of the process that requested the reconfigure and the reconfigure data.

Output

The address of the process that requested the reconfigure and the reconfigure data.

Processing

If the reconfigure command contains a new configuration file, the file will be read and the data will be validated. If the command contains a new primary/secondary *TDB* site name indicated with a "$", the *ASP* will disconnect from the *TDB* on the "old" site and attempt to register to the *TDB* on the "new" site. In either case if successful, a message is sent back to the requesting process. The primary/secondary site information issued via a reconfigure will not be maintained if the *ASP* process is restarted.

Error Conditions and Handling

If a reconfigure command is sent to a remote *ASP*, an error message will be sent to the requesting process. If the configuration file is invalid, an error message is sent to the requesting process, and the file is ignored. If the primary and/or secondary site name is invalid, an error message is sent to the requesting process, and the data is ignored.

19.1.9 process_return_message

Purpose

This procedure processes a message that was returned to *ASP* due to an error.

Input

Returned message.

Output

Returned message.

Processing

If the message was returned with an error of invalid destination address, the address is then removed from the client (*TSD*) list.

Error Conditions and Handling

All returned messages are written to the error file.

19.1.10 process_stats_lev_0, lev_1, lev_2

Purpose

These procedures process the specified statistics request.

Input

The address of the process requesting statistics.

Output

The address of the process requesting statistics.

Processing

The appropriate statistics information is obtained and sent to the requesting process. All levels of statistics contain the *ASP* process address, version number, and elapsed execution time. The following additional information is provided:

- Level 0 contains the *TDB* registration status and primary/secondary *TDB* site names if local. It also contains the current alert file name, number of elements, and other file information. It also provides a list of any registered clients.
- Level 1 contains the node switch statistics including number of messages/bytes sent and received, number of pending messages, retries, and messages lost. It also contains the number of mailbox open attempts, successes, and closes.
- Level 2 contains the registration status, primary/secondary *TDB* site names, and list of remote site/node names to which data is being sent. This applies to the local *ASP* process only.

Error Conditions and Handling

None.

19.1.11 process_tdb_message

Purpose

This procedure processes a message from the *TDB* containing the name of the alert file.

Input

Alert filename and size.

Output

None.

Processing

The date and time from the alert file is obtained and stored as the current report time. The path name of the alert file is obtained, and the file is then processed.

Error Conditions and Handling

If *ASP* is unable to get the full path name of the alert file, an error is written to the error file, and the file is ignored.

19.1.12 read_adaptation_file

Purpose

This procedure reads the *ASP* configuration file.

Input

Network Address.

Output

Network Address.

Processing

Any lines starting with a # or blank lines in the configuration files are taken as comments and ignored. All the data is read, validated, and then the appropriate parameters are set. Refer to the input section earlier in the document for the information found in the configuration file.

Error Conditions and Handling

If any errors are found in opening, reading or in the data in the configuration file, an error message is written to the trace file, and the file is ignored.

19.1.13 register_to_tdb

Purpose

This procedure registers or un-registers the *ASP* to the *TDB* for alert services depending on the value of the flag sent in the registration message.

Input

Registration flag. True = register, False = un-register.

Output

None.

Processing

The *TDBEV* address is set by obtaining the site and class identifiers. The node and invocation numbers are wild-carded to "any". *ASP* registers for "ALERT" services from the *TDB*. The toolkit registration call is made via Network Addressing. The *ASP* will wait three minutes for a successful registration message from the *TDB*. After a successful registration has been made, a ten-minute no data time-out is set.

Error Conditions and Handling

If unable to obtain the *TDBEV* class number or the site identifier, an error is written to the trace file, and the registration attempt is not made. If the registration call fails, an error is written to the trace file.

19.1.14 send_tmd_file

Purpose

This procedure sends the **tmd** file to specified sites specified in the *ASP* configuration file. This applies only to the local *ASP*.

Input

Tmd file name and size.

Output

None.

Processing

For each destination site specified in the configuration file, the file transfer message for the FTP is built which includes the time-out value and the destination *ASP* address, source and destination file name/size, priority and security values. The priority and security are constant values for the *ASP*. The site/node names from the configuration file are added to the destination file names. The remote *ASP* address is set to the specified site identifier and the node and invocation are wild-carded to "any". The toolkit file transfer call is made via Network Addressing.

Error Conditions and Handling

If the *ASP* is unable to obtain the site identifier for a specified site, the **tmd** file is not sent to the site. If the file transfer call fails due to the node switch connection, the *ASP* attempts to connect to the node switch. If the file transfer calls fails due to the node switch channel being full, the *ASP* will wait for 5 seconds and attempt the file transfer again. If the call fails for any other reason, an error is written to the trace file.

19.1.15 send_to_node

Purpose

This procedure sends a message to a process via the node switch.

Input

Message text and size. Type of message. Return flag indicating whether the message should be returned if unable to reach its destination.

Output

None.

Processing

This procedure sends a message to a specified process/processes via the Network Addressing system. It also keeps statistics on the number of messages/bytes sent to the node switch.

Error Conditions and Handling

If the send to the node switch fails due to the connection being closed, an attempt is made to reopen the node switch.

19.1.16 switch_site

Purpose

This procedure is called to switch a registration from a *TDBEV* on one site to the *TDB* on another site.

Input

None.

Output

None.

Processing

This routine switches between the primary and secondary sites for the *TDBEV*. It will close the current registration on the old and attempt a new registration on the new site. *ASP* saves which is the currently registered site. If there is no secondary site available, the switch will not occur.

Error Conditions and Handling

None.

19.1.17 wait_for_action

Purpose

This procedure is the main processing routine that waits on various events to occur.

Input

None.

Output

None.

Processing

This processing phase is an infinite loop of waiting for one of two events to occur: a message from the node switch or a timer to expire. For a timer event, the following checks are made:

- If registered to the *TDB* and no data has been received for ten minutes, the *TDB* site is switched to either primary or secondary.
- If not registered to the *TDB* and no registration acceptance has been received within three minutes, the *TDB* site is switched to either primary or secondary.
- If not currently registered to *TDB*, *ASP* will attempt another registration to the current *TDB* site.

Error Conditions and Handling

None.

19.2 The Alert Data Maintenance and Reformatting Function

The following routines are involved in the Alert Data Maintenance and Reformatting function.

19.2.1 add_element

Purpose

This procedure is used to determine where to insert an alerted element into the internal element table maintained by *ASP*.

Input

Element_id_t. Refer to Table 19-6.

Output

None.

Processing

This routine calculates the index into the linked list of alerted elements of where to insert a new element read from alert data from the *TDB*.

Error Conditions and Handling

None.

19.2.2 archive_file

Purpose

This procedure archives the alert data from the *TDB* into an ASCII format for data analysis.

Input

File stream identifier.

Output

File stream identifier.

Processing

This routine reads the **tmd** file and creates a separate archive file for each element type: airport, sector, or fix. The files are created in the /asp directory and are in the format: ammddhh.mm for airports, smmddhh.mm for sectors, and fmmddhh.mm for fixes. For each alerted element they contain the element name and type, demand, capacity, and alert status.

Error Conditions and Handling

If unable to access the **tmd** file or create the new file, the archive file is not generated.

19.2.3 create_tmd_file

Purpose

This procedure reads up to three alert files received from the *TDB* and combines them into a single file known as the **tmd** file.

Input

Tmd file path name and size.

Output

File creation status flag.

Processing

The *ASP* opens each of the alert files (airport, sector, and fix) and writes the header information and the data for each element into a record in the new **tmd** file. The seek key for each record is stored in the internal element table for future reference to the element data.

Error Conditions and Handling

If an error occurs in reading from the *TDB* files or writing to the **tmd** file, the **tmd** file will not be generated.

19.2.4 get_element

Purpose

This procedure searches for a specified alert element.

Input

Element_id_t. Refer to Table 19-6.

Output

Element_id_t. Refer to Table 19-6. Element index. If the element is not found, the index is set to -1.

Processing

This routine calculates the index into the element list for a specified element and returns the index and data for that element.

Error Conditions and Handling

None.

19.2.5 process_file

Purpose

This procedure reads an alert file received from the *TDB*.

Input

Alert file name and size. File stream ID.

Output

File stream ID.

Processing

This routine is called up to three times to process the alert files from the *TDB*. The *TDB* provides the *ASP* with the path name of one of the three files in the format: tma.mmddhhmmss, tms.mmddhhmmss, or tms.mmddhhmmss. The *ASP* will then read the other two files if available. For each file, the header and data are read for each element. Each element is added to the internal element list with the following information: element name, type, subtype, alert status, and alerted intervals.

Error Conditions and Handling

If any errors occur during file processing, an error is written to the trace file and the file is ignored

19.2.6 process_ttm_files

Purpose

This procedure oversees the creation and distribution of the **tmd** files.

Input

Alert file name and size.

Output

None.

Processing

This routine calls process_file for each of the *TDB* alert files and then create_tmd_file. It then calls send_tmd_file to transfer the **tmd** file to the remote sites. If the archive flag is set, it calls archive_file to create the files. It then calls build_map_global_time_line_msg to generate the global time line for the *TSD*s. If there are any registered clients, the alert_users routine is called to distribute the global time line files.

Error Conditions and Handling

If unable to obtain the path name of the alert file from the *TDB*, an error is written to the trace file and the file is ignored.

19.2.7 remote_process_file

Purpose

This procedure processes the received **tmd** file from the local *ASP*. It applies only to remote *ASP* processes.

Input

Tmd file name, size, and file stream identifier.

Output

Tmd file stream identifier.

Processing

This routine is called to process the **tmd** file from the hub site *ASP*. This file contains data for each element type: airports, sectors, and fixes. The header and data are read for each element of each type. Each element is added to the internal element list with the following information: element name, type, subtype, alert status, and alerted intervals.

Error Conditions and Handling

If any errors occur in opening or reading the **tmd** file, an error is written to the trace file and the file is ignored.

19.2.8 rename_tmd_files

Purpose

This procedure renames the **tmd** file at the remote *ASP* site.

Input

None.

Output

None.

Processing

This routine changes the name of the **tmd** file received from the local *ASP,* to be stored in the */sio_files* directory, by assigning a two-digit suffix to the file name. It does this by obtaining the next numeric value, getting the file path name, adding the suffix, deleting any similarly named file, and then renaming the received **tmd** file to the new suffixed format.

Error Conditions and Handling

If unable to get the path name of the received file or cannot change the name of the file, an error is written to the trace file.

19.3 The Service Provider Function

The following routines are involved in the Service Provider function.

19.3.1 alert_users

Purpose

This procedure sends the global time line information to all registered clients (users) usually *TSD*s.

Input

map_global_time_line_msg_t. Refer to Table 19-8. Number of services requested.

Output

None.

Processing

This routine walks through the linked list of registered clients and sends the global time line data to each client via the node switch. It will send up to eight addresses in one message to the node switch to reduce message traffic.

Error Conditions and Handling

None.

19.3.2 build_bar_chart

Purpose

This procedure builds a bar chart file in response to a request from a registered client.

Input

Bar chart file name and size. Element index.

Output

None.

Processing

The seek key is used to read the specified element record from the **tmd** file. The bar chart information is extracted and written to the file. Refer to Table 19-12 for the bar chart record format.

Error Conditions and Handling

If an error occurs in creating the bar chart file, an error is written to the trace file and the *ASP* process aborts. If an error occurs in reading the information from the **tmd** file, an error is written to the trace file and the bar chart file is closed.

19.3.3 build_map_element_data_msg

Purpose

This procedure builds a message for a registered client containing the bar chart information.

Input

None.

Output

None.

Processing

The get_element routine is called to find and obtain the specified element information. The build_bar_chart routine is then called to create the bar chart file. This information is then written to the element data message. The bar chart file name is: bar_chart.file_number where

file number is a unique number generated for each bar chart and global time line file. Refer to Table 19-11 for the message format.

Error Conditions and Handling

If the requested element cannot be found in the element list, no data is sent to the client.

19.3.4 build_map_global_time_line_msg

Purpose

This procedure builds a global time line file.

Input

None.

Output

None.

Processing

This routine is called whenever a new **tmd** or alert file is received. It creates the global time line file in the format: global_time_line.file_number where file number is a unique number generated for each global time line and bar chart file. It walks through the element list to extract the appropriate information. Refer to Table 19-13 for the global time line record format.

Error Conditions and Handling

If an error occurs in creating the global time line file, an error is written to the trace file.

19.3.5 process_tsd_acknowledgement

Purpose

This procedure processes acknowledgement messages from registered clients.

Input

ETMS network address. Message code, text, and size.

Output

ETMS network address. Message code, text, and size.

Processing

This message is received from clients (*TSD*s) when an element has been examined and "turned green". The *ASP* finds the specified element in the element list and updates the alert status to "G" for green. Refer to Table 19-9 for the acknowledgment message format.

Error Conditions and Handling

None.

19.3.6 process_bar_chart_request

Purpose

This procedure processes a bar chart request from a registered client.

Input

ETMS network address. Message code, text, and size.

Output

ETMS network address. Message code, text, and size.

Processing

This routine calls build_map_element data_msg, which sets up the bar chart data message and also calls build_bar_chart to create the bar chart file. It sends the requested information to the client via the node switch. Refer to Table 19-10 for the bar chart request format.

Error Conditions and Handling

None.

19.3.7 register_user

Purpose

This procedure registers a client, usually an *TSD* to the *ASP* for the specified services.

Input

ETMS network address. Registration message. Refer to Appendix A for the Network Addressing registration message format.

Output

ETMS network address. Registration message. Refer to Appendix A for the Network Addressing registration message format.

Processing

This routine adds an entry to a linked list of clients after validating the registration information. A registration accept/ok message is then sent to the requesting process via node switch. A client registers with a service of "alert" or "ALERT" with the *ASP*. Refer to Table 19-7 for the format of an entry on the client linked list

Error Conditions and Handling

If the registration message contains invalid service count or types, an registration failed/bad error message is sent to the requesting process via the node switch.

19.3.8 unregister_all_users

Purpose

This procedure disconnects all registered clients to the *ASP*.

Input

None.

Output

None.

Processing

This routine is called as part of the *ASP* cleanup handler. It walks through the list of registered clients and removes the entries from the linked list and sends a registration close message to each client.

Error Conditions and Handling

None.

19.3.9 unregister_user

Purpose

This procedure disconnects a specific client to the *ASP*.

Input

ETMS network address.

Output

ETMS network address.

Processing

This routine removes a specific entry from the linked list of registered clients.

Error Conditions and Handling

None.

19.4 Source Code Organization for the Alert Server Processor

This section describes the source code used in building the executable version of the *Alert Server Processor*. The source code resides in a Pascal file. The file contains one or more functional units called a *routine*. A routine is implemented as a Pascal function or procedure. The file has been organized as an element in a DSEE library. Before the *ASP* can be run, the following DSEE commands must be issued:

(1) **set system netaddress.sys**

(2) **set library netaddress.lib**

(3) **set model asp.sml**

(4) **set thread -default**

(5) **build**

19.5 Alert Server Processor Data Structures

The *ASP* implements data structures for three main purposes: communicating with the TDB, manipulating data internally, and communicating with the TSD. These three types of data structures are described in Sections 19.5.1 through 19.5.3.

19.5.1 Interfaces Between the ASP and the TDB

The following section describes the data items used in the *ASP* interface with the *TDB*. Records are read from the *TDB* data files into these record structures. For each data set, there are three data files, one for each element type. Each of the element files begins with a report header record, illustrated in Table 19-1. These should be identical for each element file within a data set. The report header record identifies the number of intervals, number of elements, and interval length that the data set contains. Refer to Section 26 *TDB* for more information.

The following two enumerated types (see Table 19-2) are defined:

- **NAS_element_t** - (**airport_type**, **arrival_fix_t**, **adapted_fix_type**, **navaid_type**, **low_sector_type**, **high_sector_type**, **superhigh_sector_type**, **other_type**).
- **NAS_event_t - (airport_departure, airport_arrival, low_fix_crossing, high_fix_crossing, superhigh_fix_crossing, sector_crossing).**

Table 19-1. report_hdr_rec_t Data Structure

report_hdr_rec_t	
Library Name: netaddress.lib	**Purpose:** To hold header information for reports generated by the TDB. This is the first record of all TDB record-structured reports.

Element Name: record.files.ins.pas				
Data Item	**Definition**	**Unit/Format**	**Range**	**Var. Type/Bits**
software_version	Version number of this report	currently 1.0		real
report_start	Start date and time for the period covered in this report.	system type		cal_$ timedat_rec_t
report_end	End date and time for the period covered in this report.	system type		cal_$ timedat_rec_t
report_toc	Record containing additional header information. See	record of integers		traffic_toc_rec_t
number_of_elements	Number of elements contained in this report.			integer
number_of_intervals	Number of intervals of data for each element.			integer
interval_length	Number of minutes making up an interval.	currently 15 min		integer

Next in the element files are the elements. The data for each element consist first of an element header record (see Table 19-2) For each interval in the data set, there is one or more count records (see Table 19-3). The exact number depends upon the sub-type within the element. Airports have two: departures and arrivals. Fixes have three: low, high, and super-high. Sectors have a single sub-type.

Table 19-2. element_hdr_rec_t Data Structure

element_hdr_rec_t				
Library Name: netaddress_lib		**Purpose:** To hold element header information for each element in a TDB record-structured report.		
Element Name: record.files.ins.pas				
Data Item	**Definition**	**Unit/Format**	**Range**	**Var. Type/Bits**
element_name	The NAS element name for this element.	array of 10 characters		NAS_element_name_t
element_type	The type of element in this record.	enumerated type		NAS_element_t
NAS_evern_set	Set of event kinds in this report. Corresponds to element type.	set of enumerated		NAS_event_set_t
alert_status	Color management field for alert status. Original status color.	R = Active alert Y = Proposed	Q,R,X,Y,G	char
color_char	Color management field assigned by traffic management.	G = No alert	Q,R,X,Y,G	char

Table 19-3. count_rec_t Data Structure

count_rec_t				
Library Name: netaddress_lib		**Purpose:** To hold statistics information as part of a TDB report-structured report. The statistics are for a single element type, element kind, interval triplet.		
Element Name: record.files.ins.pas				
Data Item	**Definition**	**Unit/Format**	**Range**	**Var. Type/Bits**
nominal_cap	Default capacity value.	flight count	0 – 255	char
today_cap	Today's capacity value.	flight count	0 – 255	char
nominal_ga	Default GA estimate value.	flight count	0 – 255	char
today_ga	Today's GA estimate value.	flight count	0 – 255	char
ga_cont	Number of GA flights.	flight count	0 – 255	char
scheduled_count	Number of scheduled flights.	flight count	0 – 255	char
active_count	Number of active flights	flight count	0 – 255	char
total_count	Total number of flights.	flight count	0 – 255	char

flight_count	Number of flights actually in the flight list.	flight count	0 – 255	char
alert_status	Indication whether this element is alerted	R = act, Y = pro, G = no alert	Q,R,X,Y,G	char

19.5.2 Internal ASP Data Structures

The data structures described in Tables 19-4 through 19-7 are used internally in the *ASP*. The data is read from the *TDB* element files. The element files (up to three) are combined into a single **tmd** file. This file can then be easily transmitted to a remote *ASP*.

The internal *ASP* data structure starts with a header record (see Table 19-4), followed by the **element_data** (see Table 19-5) and **element_id_t** (see Table 19-6) records.

The *ASP* maintains a linked list (see Table 19-7) of all processes registered to it.

Table 19-4. new_report_hdr_rec_t Data Structure

new_report_hdr_rec_t				
Library Name: netaddress.lib		**Purpose:** To hold header information for reports generated by the TDB.		
Element Name: asp.pas				
Data Item	**Definition**	**Unit/Format**	**Range**	**Var. Type/Bits**
software_version	Version number of this report	currently 1.0		real
report_start	Start date and time for the period covered in this report.	system type		cal_$ timedat_rec_t
report_end	End date and time for the period covered in this report.	system type		cal_$ timedat_rec_t
report_time	Date and time for the report file name.	system type		cal_$ timedat_rec_t
report_toc	Record containing additional header information. See below.	record of integers		traffic_toc_rec_t
number_of_ elements	Number of elements contained in this report.			integer
number_of_intervals	Number of intervals of data for each element.			integer
interval_length	Number of minutes making up an interval.	currently 15 min		integer

Table 19-5. element_data Data Structure

element_data				
Library Name: netaddress.lib		**Purpose:** Table of current element data.		
Element Name: asp.pas				
Data Item	**Definition**	**Unit/Format**	**Range**	**Var. Type/Bits**
element_data	Array of records containing alerted element information.	Array of element_id_t		Array of records

Table 19-6. element_id_t Data Structure

element_id_t				
Library Name: neladdress.lib		**Purpose:** To contain alerted element information.		
Element Name: asp.pas				
Data Item	**Definition**	**Unit/Format**	**Range**	**Var. Type/Bits**
element_type	Element type from element_hdr_rec_t	Enumerated type		NAS_element_t
subtype	Type of alerted element.	Type of airport or fix †	1,2,3	char
color	Alert color for element.		Q,R,X,Y,G	char
name	Element name.			char10_t
alarmed_list	Alert colors for each interval.		1..10	array of char
filename	Numbers for filenames of bar charts.		1..5	array of integer
seek_key	Seek key for direct access to data file.			integer32.

† Airport (1 = dep 2 = arr)
Fix (1 = low 2 = high 3 = superhigh)

Table 19-7. user_list_t Data Structure

user_list_t				
Library Name: netaddress.lib		**Purpose:** To contain information on all registered processed to *ASP*.		
Element Name: asp.pas				
Data Item	**Definition**	**Unit/Format**	**Range**	**Var. Type/Bits**
next	Link to next user in linked list.			integer
address	Network address of process.	toolkit		net$_address_t
User	Registered user information.	toolkit		net$_user_reg_with_provider_t

19.5.3 Interfaces Between the ASP and the TSD

These data structures and items are used for the interface with the *TSD*. The major portion of the interface is a request/reply message operation sent via Network Addressing.

The *TSD* sends a request consisting of a message with an action code of **1** or **2**. The *ASP* satisfies the request and sends a reply message with an action code of **16001** or **16003**.

The first message **map_global_time_line_msg_t** is not a reply to any request made by the *TSD*; it is a signal that a new data set has been received and is not active. This message is also sent to the *TSD* when it first registers to the *ASP*. It contains the information necessary for the *TSD* to display the global time line (See Table 19-8).

Table 19-8. map_global_time_line_msg_t Data Structure

map_global_time_line_msg_t				
Library Name: netaddress.lib		**Purpose:** To contain information on the alerted elements to be sent to the *TSD*.		
Element Name: asp.pas				
Data Item	**Definition**	**Unit/Format**	**Range**	**Var. Type/Bits**
action_code	Code defining this message type for *TSD* interpretation.	16001 = code	16001	integer
report_time	Date/Time the alert files were created.	System type		cal_$timedate_rec_t
start_time	Alert file start time.	System type		cal_$timedate_rec_
interval_length	Number of minutes in each interval.	Minutes		integer
interval_count	Number of intervals in alert file.			integer
tmd_file_len	Alert filename size.			integer
tmd_file_name	Name of the alert file			string
global_file_len	Global tine line filename size.			integer
global_file_name	Name of global time line file.			string

The next message is sent by the *TSD* to provide a color change to a particular element. The **asp_element_acknowledgment_msg_t** message defines the element name and type (See Table 19-9).

Table 19-9. asp_element_acknowledgement_msg_t Data Structure

asp_element_acknowledgement_msg_t				
Library Name: netaddress.lib		**Purpose:** To contain a color change request from the *TSD*.		
Element Name: asp.pas				
Data Item	**Definition**	**Unit/Format**	**Range**	**Var. Type/Bits**
action_code	Code defining this message type from the *TSD*.	1 = code	1	integer
element_name	Name of element.			char10_t
element_type	Element type (sector, airport, fix)	A=airport, L,H,S =sector,	A,L,H,S,l,h,s	char

The next message is sent by the *TSD* to request a bar chart for a particular element (see Table 19-10). The element name, element type, and request type are defined.

Table 19-10. asp_element_give_me_data_msg_t Data Structure

asp_element_give_me_data_msg_t	
Library Name: netaddress.lib	**Purpose:** To contain a data request form the *TSD*.
Element Name: asp.pas	

Data Item	Definition	Unit/Format	Range	Var. Type/Bits
action_code	Code defining this message type from the *TSD*.	2 = code	2	integer
element_name	Name of the element.			char10_t
data_type	Type of data request.	B = bar chart	B	char
element_type	Element Type	A=airport, L,H,S =sector,	A,L,H,S,l,h,s	char

The **map_element_data_msg_t** message (see Table 19-11) is sent in response to a *TSD* data request. When a request is made, a file is created which contains the information request. This file is saved so that it need not be re-created when a similar request comes in from another *TSD*. Then this message is sent informing the requesting *TSD* of this file name.

Table 19-11. map_element_data_msg_t Data Structure

map_element_data_msg_t				
Library Name: etms_lib		**Purpose:** To contain the data requested by the *TSD*.		
Element Name: asp.pas				
Data Item	**Definition**	**Unit/Format**	**Range**	**Var. Type/Bits**
action_code	Code defining this message type for *TSD* interpretation.	16003 = code	16003	integer
file_name_len	Length of data filename.			integer
element_name	Name of the element.			char10_t
data_type	Type of request.	B = bar chart	B	char
file_name	Name of the data file.			string

The **bar_chart_record** structure is used in the bar chart file sent to the *TSD* when a bar chart request is made. The file contains a single record of this type. The record is a variant record, depending upon the element type: airport, fix, or sector. (See Table 19-12 for details.)

Table 19-12. bar_chart_record Data Structure

bar_chart_record					
Library Name: etms_lib		**Purpose:** To contain information necessary for the TSD to generate a bar chart.			
Element Name: asp.pas					
Data Item		**Definition**	**Unit/Format**	**Range**	**Var. Type/Bits**
interval_start_time		Alert file start time.	System type		cal_$timedate_rec_
element_type		Type of element.	Enumerated		NAS_element_t
element_name		Name of element.			char10_t
For Airports:					
	arrival_capacities	Airport arrival capacities for each interval.		1..10	array of integer
	departure_capacitie	Airport departure capacities for each interval.		1..10	array of integer
	active_arrivals	Number of airport active arrivals for each interval.		1..10	array of integer
	active_departures	Number of airport active departures for each interval.		1..10	array of integer
	total_arrivals	Airport total arrivals for each interval.		1..10	array of integer
	total_departures	Airport total departures for each interval.		1..10	array of integer
For Fixes:					
	low_capacities	Low fix capacities for each interval.		1..10	array of integer
	high_capacities	High fix capacities for each interval.		1..10	array of integer
	superhi_capacities	Superhigh fix capacities for each interval.		1..10	array of integer
	active_lo	Number of active low fix flights for each interval.		1..10	array of integer
	active_high	Number of active high fix flights for each interval.		1..10	array of integer
	active_superhi	Number of active superhigh fix flights for each interval.		1..10	array of integer
	total_low	Total number of low fix flights for each interval.		1..10	array of integer
	total_high	Total number of high fix flights for each interval.		1..10	array of integer
	total_superhi	Total number of superhi fix flights for each interval.		1..10	array of integer

Table 19-12. bar_chart_record Data Structure (continued)

bar_chart_record					
For Sectors:					
	capacities	Sector capacities for each interval.		1..10	array of integer
	active_peaks	Sector active peaks for each interval.		1..10	array of integer
	total_peaks	Sector total peaks for each interval.		1..10	array of integer

The **global_rec** structure (see Table 19-13) is used in the global time line file. One record exists for each element in the data file. This is used by the *TSD* to display the alerted elements and their colors.

Table 19-13. global_rec Data Structure

global_rec				
Library Name: etms_lib		**Purpose:** To contain the alerted element information.		
Element Name: asp.pas				
Data Item	**Definition**	**Unit/Format**	**Range**	**Var. Type/Bits**
color	Alerted element color.		Q,R,X,Y,G	char
e_type	Element type.		A,L,H,S,l,h,s	char
name	Name of the element.			char10_t
alarms	Alerted intervals for the element.		1..11	char
alarms_color	Color for alerted intervals for the element.		Q,R,X,Y,G	char